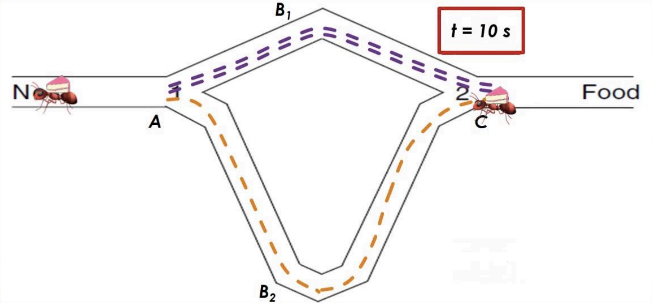
Ant Colony Optimization Algorithms

Tan Ningyuan, Zhong Yuhui, Lu Liangrui.

2019/05/31

Instruction

- Ant colony optimization is a technique for optimization that was introduced in the early 1990's
- The inspiring source of ant colony optimization is the foraging behaviour of real ant colonies



Instruction

- It is based on a study of the collective foraging behavior of real ant colonies in nature, simulating the real ant colony collaboration process.
- The algorithm constructs a solution path by several ants, and improves the quality of the solution by leaving and exchanging pheromones on the solution path to achieve the purpose of optimization.

Data Set Parser

- The TSP problem is a very classic optimization problem.
- There are many online data
- http://www.math.uwaterloo.ca/tsp/data/index.html
- We choose real data so that the results can be more practical.
- The format of these data is different.
- So The first part of our program is the standardized data format.

Data standardization

We convert the form of the coordinates into a matrix form. The data structure used is Eigen

Data

Argentina - 9,152 Cities (Includes duplications; 6,723 distinct cities)
Point Set
Tour (within 0.012% of optimal)
CIA World Factbook Map
Data (TSPLIB Format, ar9125)
Log of Computation

Burma - 33,708 Cities
Point Set
Tour (within 0.031% of optimal)
CIA World Factbook Map
Data (TSPLIB Format, bm33708)
Log of Computation

China - 71,009 Cities
Point Set
Tour (within 0.024% of optimal)
CIA World Factbook Map
Data (TSPLIB Format, ch71009)
Log of Computation

Djibouti - 38 Cities
Point Set
Optimal Tour
CIA World Factbook Map
Data (TSPLIB Format, dj38)
Log of Computation

Egypt - 7,146 Cities
Point Set
Tour (within 0.021% of optimal)
CIA World Factbook Map
Data (TSPLIB Format, eg7146)
Log of Computation

Finland - 10,639 Cities
Point Set
Optimal Tour
CIA World Factbook Map
Data (TSPLIB Format, fi10639)
Log of Computation

Data Format

```
NAME : eil51
COMMENT : 51-city problem (Christofides/Eilon)
TYPE : TSP
DIMENSION : 51
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
1 37 52
2 49 49
3 52 64
4 20 26
5 40 30
6 21 47
7 17 63
8 31 62
9 52 33
10 51 21
11 42 41
12 31 32
13 5 25
```

Algorithmic

$$p_{i,j}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}]^{\alpha}[\eta_{is}]^{\beta}}, & j \in allowed_k \\ 0 & otherwise \end{cases} \quad (1)$$

Algorithmic Detail

α the parameter to regulate the influence of τ_{ij}, η_{ij} the visibility of city j from city i, which is always set as $1/d_{ij}$ (d_{ij} is the distance between city i and j), β the parameter to regulate the influence of η_{ij} and $allowed_k$ the set of cities that have not been visited yet, respectively.

Algorithmic

$$\tau_{i,j}(t+1) = \rho\tau_{i,j}(t) + \Delta\tau_{i,j} \tag{2}$$

$$\tau_{i,j} = \Sigma_{k=l}^{l}\Delta\tau_{i,j}^{k} \tag{3}$$

$$\tau_{i,j}^{k} = Q/L_k \tag{4}$$

Algorithmic Detail

t is the iteration counter, $\rho \in [0, 1]$ the parameter to regulate the reduction of $\tau_{i,j}$, $\Delta\tau_{i,j}$ the total increase of trail level on edge (i, j) and $\Delta\tau_{i,j}^{k}$ the increase of trail level on edge (i, j) caused by ant k, respectively.

Algorithmic

Algorithm 1 The ACS algorithm

procedure SET INIT INFORMATION ▷
    for $t = 1$ to iteration number **do**
        for $k = 1$ to l **do**
            Repeat until ant k has completed a tour
            Select the city j to be visited next
            With probability p_{ij} given by Eq.(1)
            Calculate L_k
            Update the trail levels according to Eqs(2-4).
        end for
    end for
end procedure

Algorithmic

Algorithm 2 CONSTRUCT ROUTES

procedure CONSTRUCT ROUTES ▷
    for $i = 1$ to $V - 1$ **do**
        for $\forall k \in M$ **do**
            Choose the next city s_k according to the formula mentioned
            add $edge(r_k, s_k)$ to $Tour_k$
            $r_k = s_k$
        end for
    end for
    for $\forall k \in M$ **do**
        add $edge(r_k, r_{k1})$ to $Tour_k$
    end for
end procedure

File Tree

```
.
├── AntColonyBase.cpp
├── CMakeLists.txt
├── README.md
├── cmake
│   ├── get_eigen.cmake.in
│   └── misc.cmake
├── data
│   ├── eil51.opt.tour
│   ├── eil51.tsp
│   ├── p01.tsp
│   ├── ulysses16.opt.tour
│   └── ulysses16.tsp
├── include
│   └── AntColonyBase.h
└── test
    ├── CMakeLists.txt
    └── test.cpp
```

AntColonyBase.h

```cpp
// the contents of a AntColonyBase.
// Example:
//    AntColonyBase = data(argv[1]);
//    data.calcTSP();
//    std::deque<int> &path = data.get_path();


class AntColonyBase
{
 public:
  explicit AntColonyBase(const char *filename, double alpha = 15, double beta = 20,
      double rho = 0.1, double colony_eff = 1.0, unsigned maxiter = 500);
  explicit AntColonyBase(const std::string &filename, double alpha = 15, double beta = 20,
                         double rho = 0.1, double colony_eff = 1.0, unsigned maxiter = 500);
  AntColonyBase(const AntColonyBase&) = delete;
  AntColonyBase &operator=(const AntColonyBase&) = delete;
  int calcTSP();
  int recalcTSP();
  std::deque<int> &get_path();
  std::deque<double> &get_mintour_each();
  std::deque<double> &get_mintour_global();
  void printAdj(std::ostream &os);
  double total_len();
 private:
  enum NODE_COORD_TYPE {...};
  struct Point {...};
  double _alpha;                // Regulate the influence of the intensity of pheromone
  double _beta;                  // Regulate the influence of visibility of city
  double _rho;                    //Rate at which each pheromone disappears
  double _colony_eff;
  unsigned _maxiter;

  int _dim;
  Eigen::MatrixXd _adj_matrix;
  bool _caculated;
  std::deque<int> _path;
  std::deque<double> _mintour_each;
  std::deque<double> _mintour_global;
  std::string error_msg;
};
```

Std::discrete-distribution

- std::discrete-distribution produces random integers on the interval [0,n), where the probability of each individual integer i is defined as w i/S, that is the weight of the ith integer divided by the sum of all n weights.
- std::discrete-distribution satisfies all requirements of RandomNumberDistribution

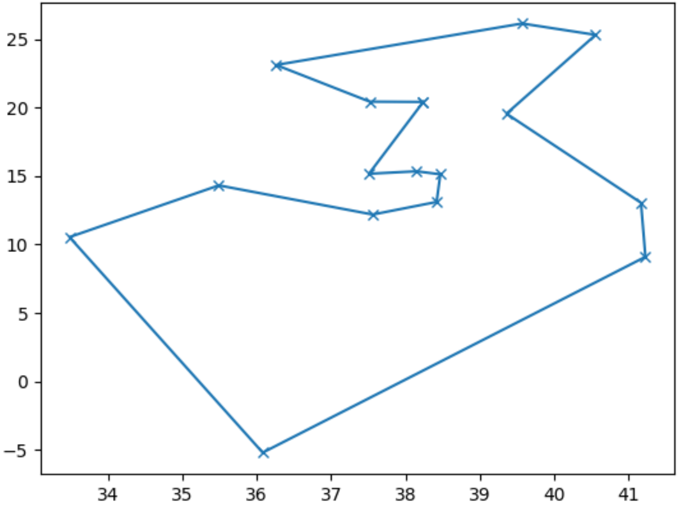
Multithreading

- Compared with some other algorithms, the advantage of ant colony algorithm is that it can run in multiple threads
- We also implement it by Using OpenMP

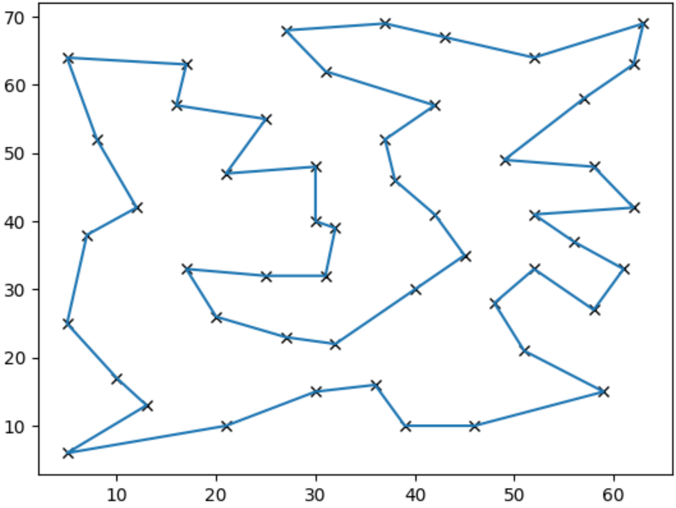
Evaluate

- Evaluate the correctness, because the TSP problem is an NP problem, and the ant colony algorithm is a heuristic algorithm. It is not guaranteed to be an optimal solution, so we will evaluate the correctness.

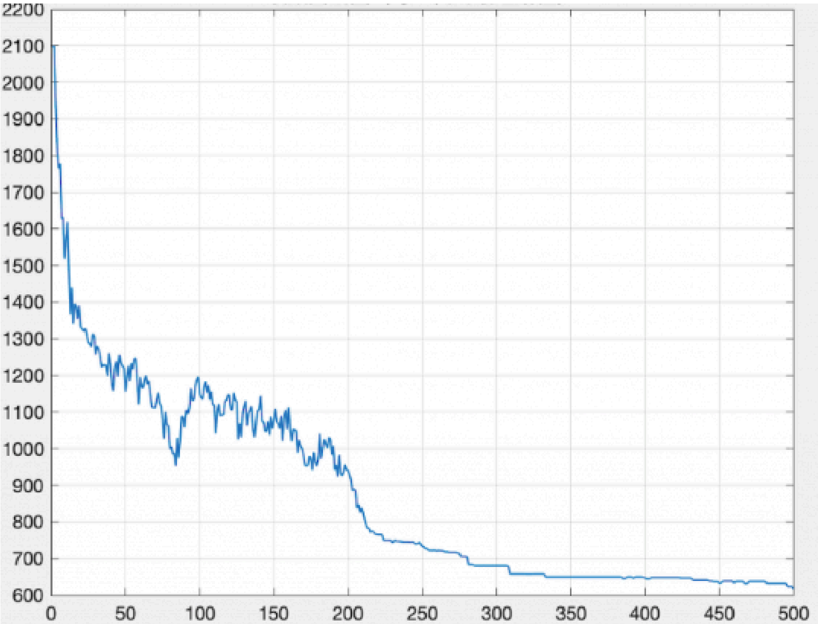
Result



Result



Result



2200
2100
2000
1900
1800
1700
1600
1500
1400
1300
1200
1100
1000
900
800
700
600
0
50
100
150
200
250
300
350
400
450
500

Conclusion

- It can be seen that when the number of cities is moderate, the iteration 500 shortest path length has a tendency to converge.
- If the parameters α and β are set improperly, the solution speed is very slow and the quality of the obtained solution is particularly poor.
- The basic ant colony algorithm has a large amount of calculation, and the solution takes a long time.